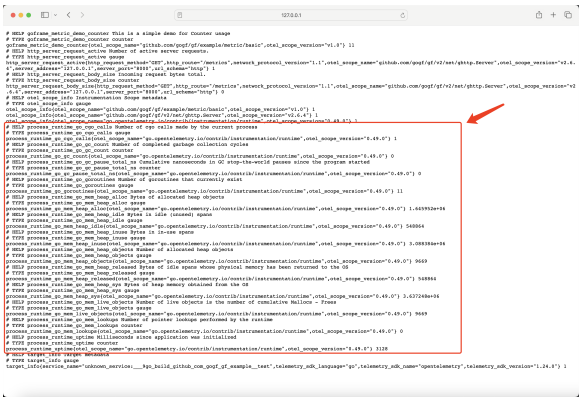
-

GoProviderotelmetric.WithBuiltInMetrics()

```
package main

import (
	"go.opentelemetry.io/otel/exporters/prometheus"

	"github.com/gogf/gf/contrib/metric/otelmetric/v2"
	"github.com/gogf/gf/v2/frame/g"
	"github.com/gogf/gf/v2/os/gctx"
	"github.com/gogf/gf/v2/os/gmetric"
)

const (
	instrument        = "github.com/gogf/gf/example/metric/basic"
	instrumentVersion = "v1.0"
)

var (
	meter = gmetric.GetGlobalProvider().Meter(gmetric.MeterOption{
		Instrument:        instrument,
		InstrumentVersion: instrumentVersion,
	})
	counter = meter.MustCounter(
		"goframe.metric.demo.counter",
		gmetric.MetricOption{
			Help: "This is a simple demo for Counter usage",
			Unit: "bytes",
		},
	)
)

func main() {
	var ctx = gctx.New()

	// Prometheus exporter to export metrics as Prometheus format.
	exporter, err := prometheus.New(
		prometheus.WithoutCounterSuffixes(),
		prometheus.WithoutUnits(),
	)
	if err != nil {
		g.Log().Fatal(ctx, err)
	}

	// OpenTelemetry provider.
	provider := otelmetric.MustProvider(
		otelmetric.WithReader(exporter),
		otelmetric.WithBuiltInMetrics(),
	)
	provider.SetAsGlobal()
	defer provider.Shutdown(ctx)

	// Counter.
	counter.Inc(ctx)
	counter.Add(ctx, 10)

	// HTTP Server for metrics exporting.
	otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
}
```

http://127.0.0.1:8000/metrics

Content Menu



process.runtime.go.cgo.calls	Counter		cgo
process.runtime.go.gc.count	Counter		gc
process.runtime.go.gc.pause_ns	Histogram	ns	GC stop-the-world
process.runtime.go.gc. pause_total_ns	Counter	ns	GC stop-the-world
process.runtime.go.goroutines	Gauge		
process.runtime.go.lookups	Counter		
process.runtime.go.mem. heap_alloc	Gauge	bytes	
process.runtime.go.mem. heap_idle	Gauge	bytes	
process.runtime.go.mem. heap_inuse	Gauge	bytes	
process.runtime.go.mem. heap_objects	Gauge		
process.runtime.go.mem. live_objects	Gauge		(Mallocs - Frees)
process.runtime.go.mem. heap_released	Gauge	bytes	
process.runtime.go.mem.heap_sys	Gauge	bytes	
process.runtime.uptime	Counter	ms	